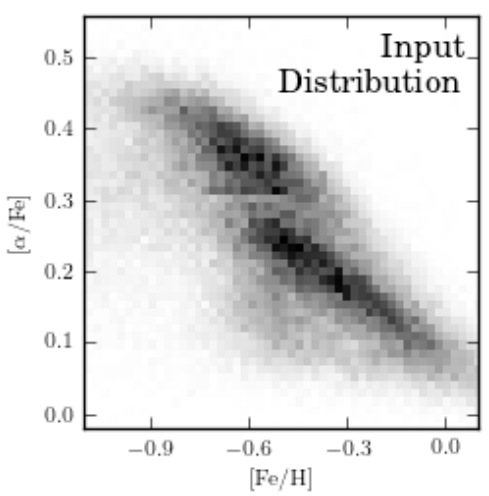
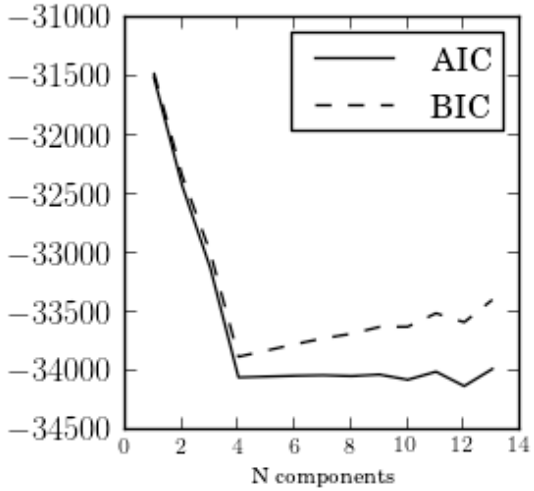
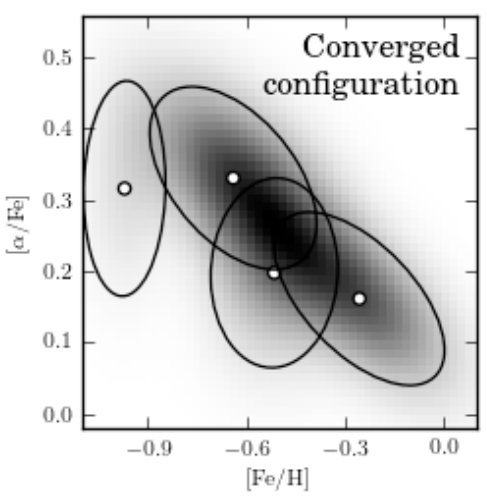
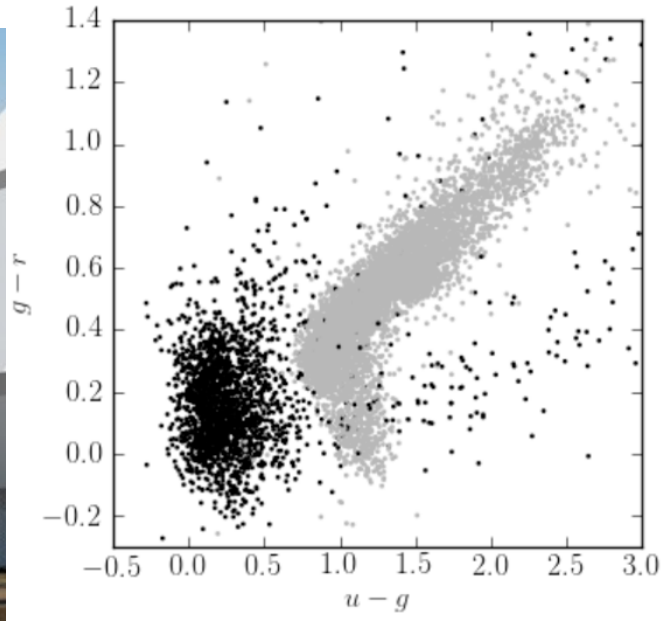
Week 9: Unsupervised classification (= clustering)

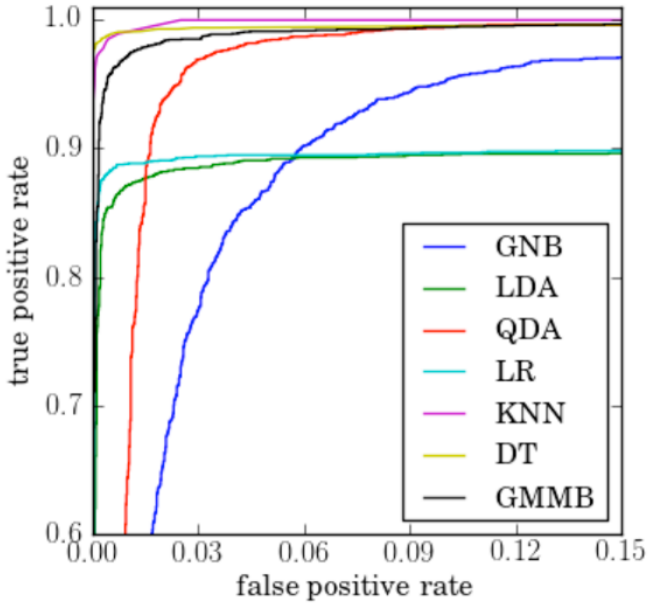
Darko Jevremović











Outline

- Clustering (unsupervised classification)
  - 1-D hypothesis testing
  - clustering with Gaussian Mixture models (GMM)
  - K-means clustering algorithm
  - hierarchical clustering algorithm
- (supervised) Classification
  - potpourri of supervised classification methods:

    naive Bayes, quadratic discriminant analysis, GMM, KNN, support vector machines
  - classification comparison with ROC curves

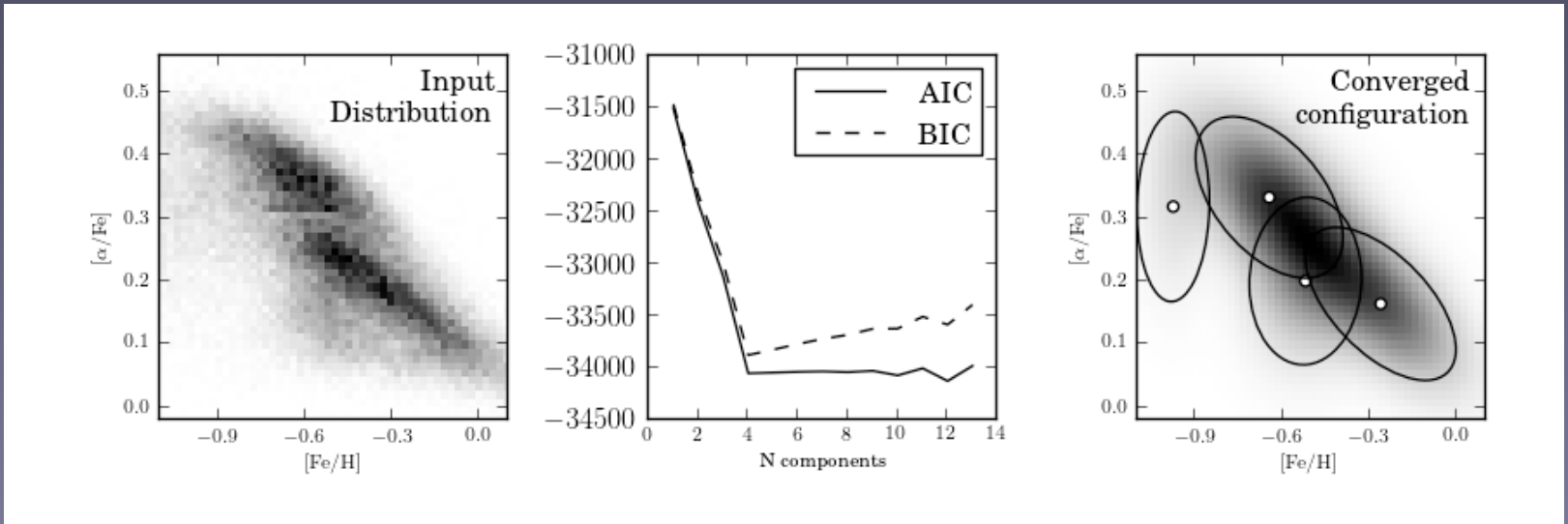
Clustering (unsupervised classification)

Here “unsupervised” means that there is no prior information about the number and properties of clusters.

“Clustering” in astronomy refers to a number of different aspects of data analysis.

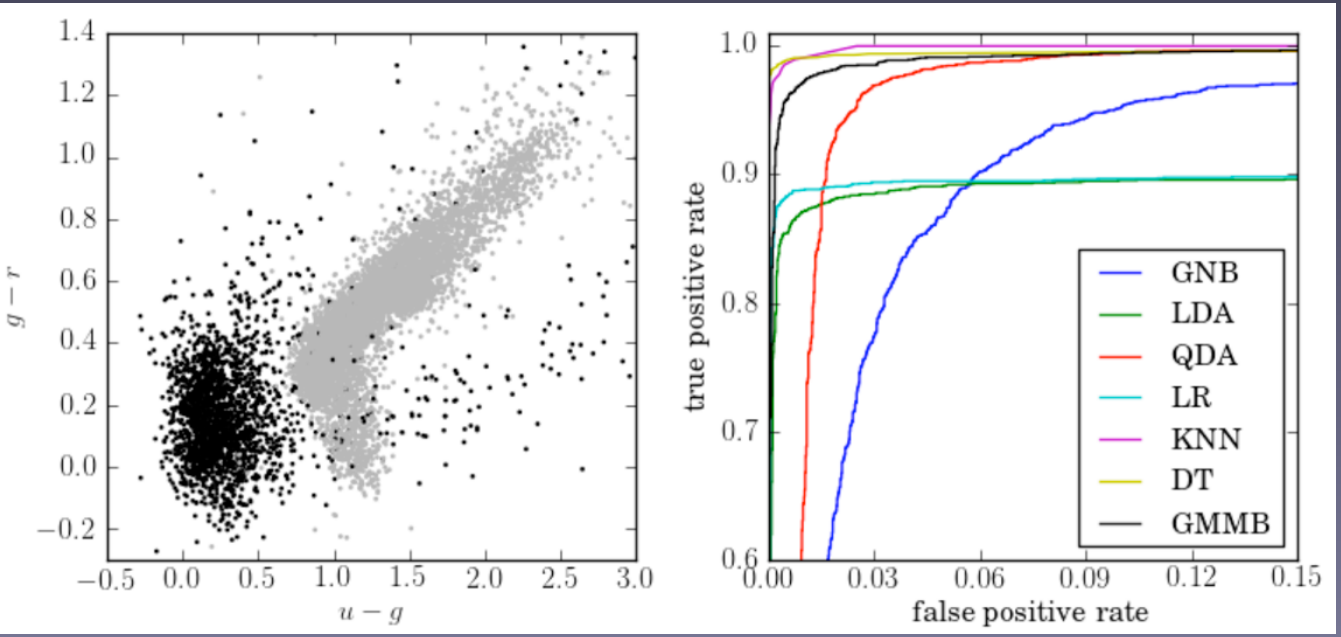
Given a multivariate point data set, we can ask whether it displays any structure, that is, concentrations of points. Alternatively, when a density estimate is available we can search for “overdensities.”

Another way to interpret clustering is to seek a partitioning or segmentation of data into smaller parts according to some criteria.



Supervised classification

Here “supervised” means that there is prior information about the number and properties of clusters: for a training sample, we know the so-called “class labels” (for each data point in the training sample, we know to which cluster it belongs; characterizing these known clusters is typically easier than finding and characterizing unknown clusters)



A quick question:

You just measured x = 3, with a negligible measurement error.

You know that you could have drawn this value from one of two possible populations (e.g. stars and galaxies). One population can be described as N(0,1), and the other one as N(4,1).

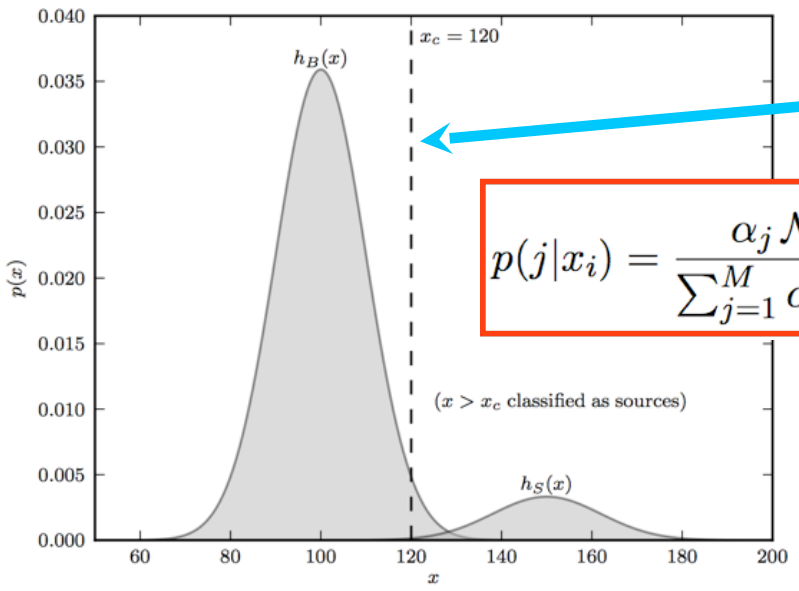
Which population is more likely, given your x?

1-D hypothesis testing

Often the underlying distribution from which data $\{x_i\}$ were drawn, $h(x)$, is a sum of two populations

$$h(x) = (1 - a)\, h_B(x) + a\, h_S(x), \tag{4.36}$$

where a is the relative normalization factor (we assume that integrals of h_B and h_S are normalized to unity). In this example there is not only a null hypothesis (B, for background), but also a specific alternative hypothesis (S, for source). Given $\{x_i\}$, for example counts obtained with a measuring apparatus, we want to assign to each individual measurement x_i the probability that it belongs to population S, $p_S(x_i)$ (of course, $p_B(x_i) = 1 - p_S(x_i)$ as there are only these two



$$p(j|x_i) = \frac{\alpha_j \mathcal{N}(\mu_j, \sigma_j)}{\sum_{j=1}^{M} \alpha_j \mathcal{N}(\mu_j, \sigma_j)}$$

Figure 4.5.: An example of a simple classification problem between two Gaussian distributions. Given a value of x, we need to assign that measurement to one of the two distributions (background vs. source). The cut at $x_c = 120$ leads to very few Type II errors (i.e., *false negatives*: points from the distribution h_S with $x < x_c$ being classified as background), but this comes at the cost of a significant number of Type I errors (i.e., *false positives*: points from the distribution h_B with $x > x_c$ being classified as sources).

Key point: we need to choose this boundary to assign data x to set B or S

Often, we assign probabilistic classification

We can have more than just two mixture components

1-D hypothesis testing

If we choose a classification boundary value x_c, then the *expected number* of spurious sources (false positives or Type I errors) in the classified sample is

$$n_{\rm spurious} = N\,(1-a)\,\alpha = N\,(1-a)\int_{x_c}^{\infty} h_B(x)\,dx, \tag{4.37}$$

and the number of missed sources (false negatives or Type II errors) is

$$n_{\rm missed} = N\,a\,\beta = N\,a\int_{0}^{x_c} h_S(x)\,dx. \tag{4.38}$$

The number of instances classified as a source, that is, instances when the null hypothesis is rejected, is

$$n_{\rm source} = N\,a\, - n_{\rm missed} + n_{\rm spurious}. \tag{4.39}$$

The sample *completeness* (also called sensitivity and recall rate in the statistics literature) is defined as

$$\eta = \frac{Na - n_{\rm missed}}{Na} = 1 - \int_{0}^{x_c} h_S(x)\,dx,$$

with $0 \le \eta \le 1$, and the sample *contamination* is defined as

$$\epsilon = \frac{n_{\rm spurious}}{n_{\rm source}},$$

with $0 \le \epsilon \le 1$ (the $1-\epsilon$ rate is sometimes called classification efficiency). The sample contamination is also called the *false discovery rate* (FDR). As x_c increases, the sample contamination decreases (good), but at the same time completeness decreases too (bad). This trade-off can be analyzed using the so-called *receiver operating characteristic* (ROC) curve which typically plots the fraction of true positives vs. the fraction of true negatives (see HTF09). In astronomy, ROC curves are often plotted as expected completeness vs. contamination (or sometimes efficiency) rate.

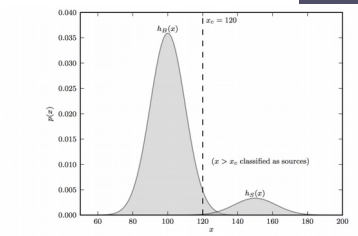


Figure 4.5.: An example of a simple classification problem between two Gaussian distributions. Given a value of x, we need to assign that measurement to one of the two distributions (background vs. source). The cut at $x_c = 120$ leads to very few Type II errors (i.e., *false negatives*: points from the distribution h_S with $x < x_c$ being classified as background), but this comes at the cost of a significant number of Type I errors (i.e., *false positives*: points from the distribution h_B with $x > x_c$ being classified as sources).

1-D hypothesis testing

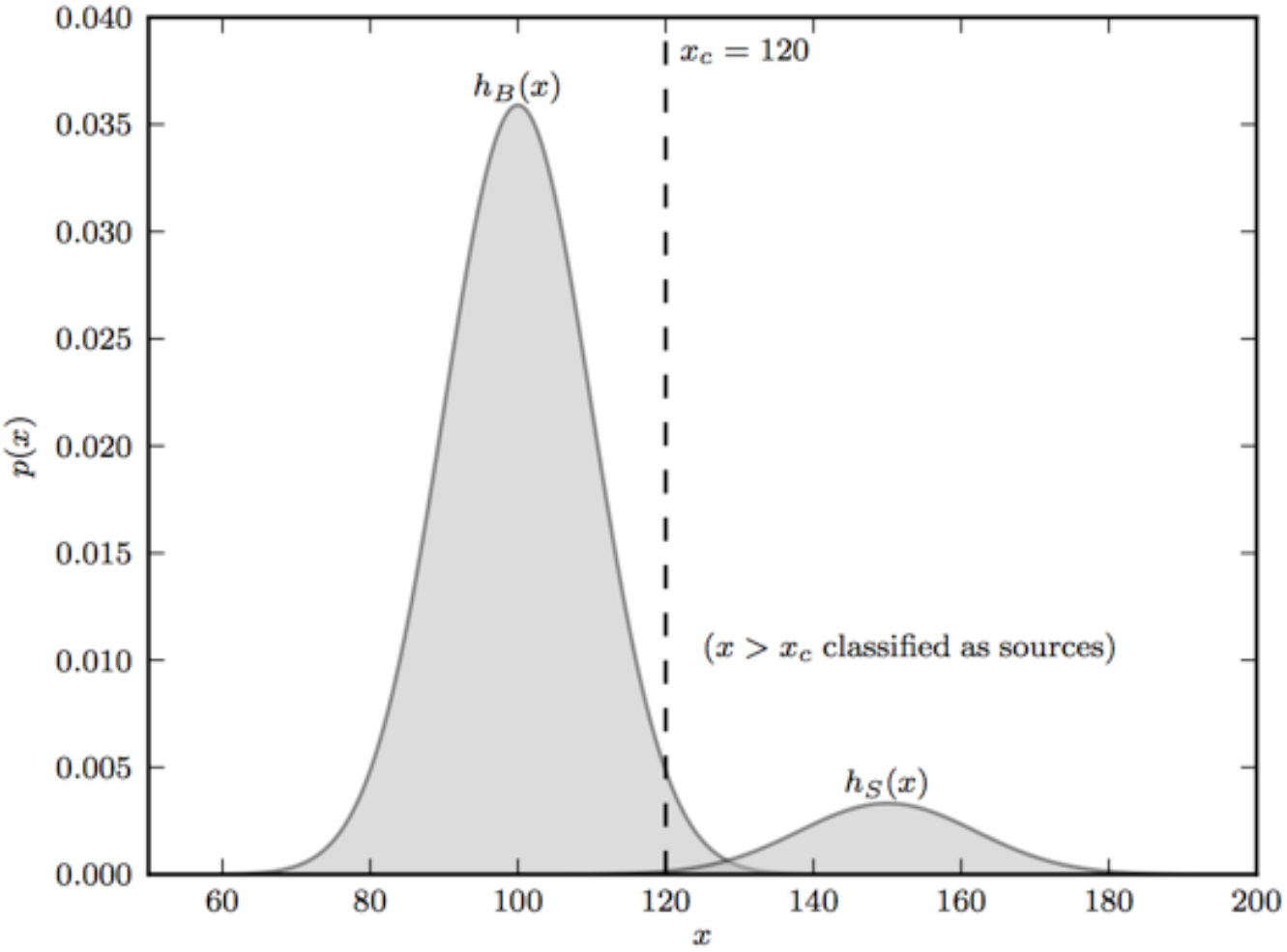


Figure 4.5.: An example of a simple classification problem between two Gaussian distributions. Given a value of x, we need to assign that measurement to one of the two distributions (background vs. source). The cut at $x_c = 120$ leads to very few Type II errors (i.e., *false negatives*: points from the distribution h_S with $x < x_c$ being classified as background), but this comes at the cost of a significant number of Type I errors (i.e., *false positives*: points from the distribution h_B with $x > x_c$ being classified as sources).

Decision boundary

The *decision boundary* between two classes is the set of x values at which each class is equally likely; that is,

$$\pi_1 p_1(\mathbf{x}) = \pi_2 p_2(\mathbf{x}); \tag{9.14}$$

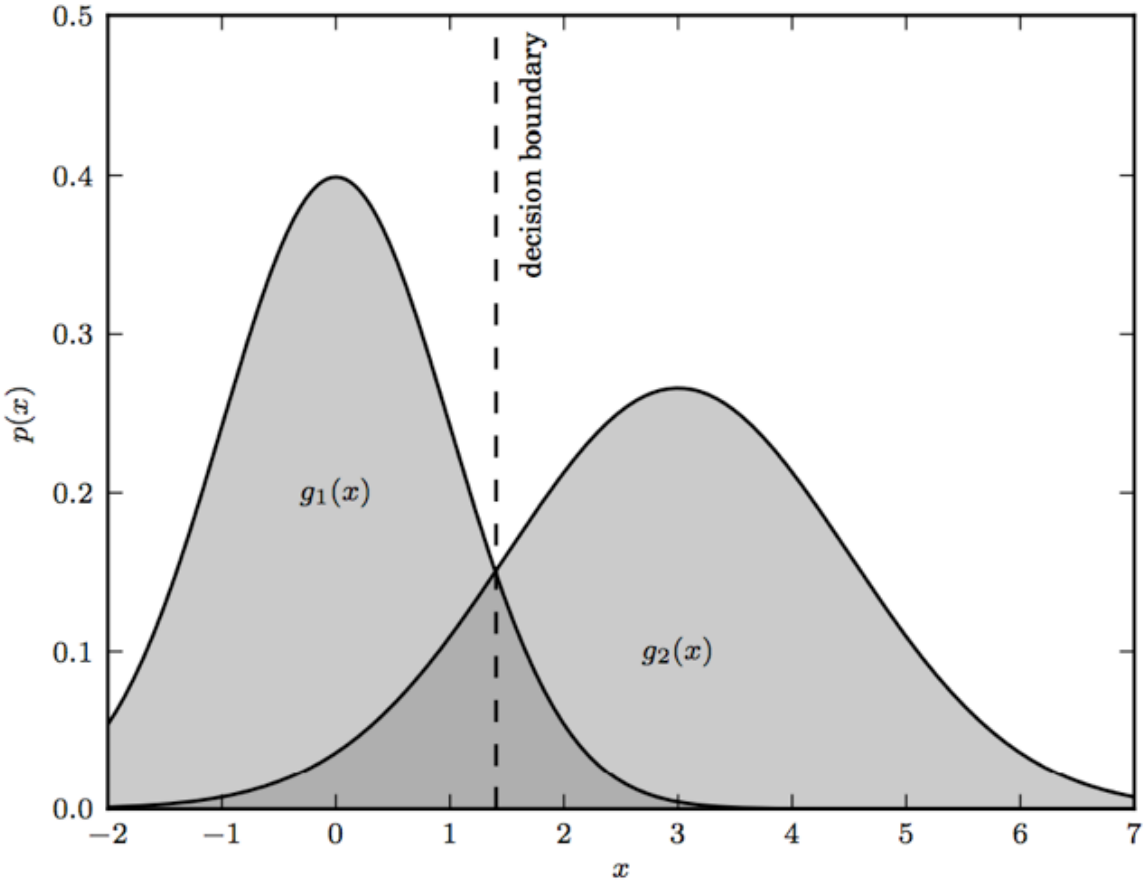


Figure 9.1.: An illustration of a decision boundary between two Gaussian distributions.

data: x=3;

p1=N(0,1)

p2=N(4,1)

you must know

priors to get

g1 and g2

from p1 and p2!

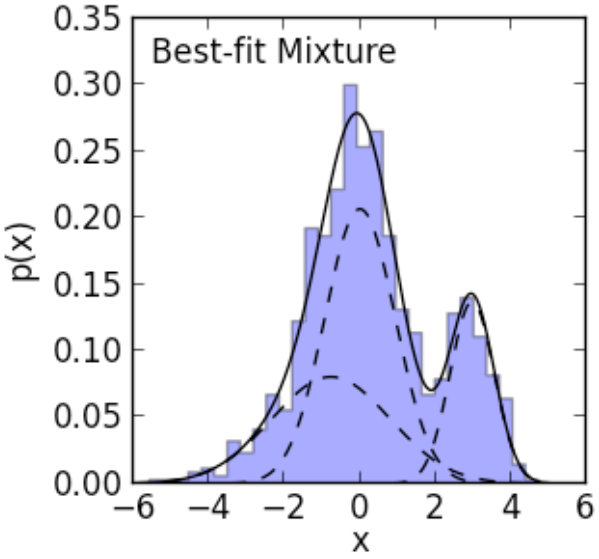
Gaussian Mixture Models

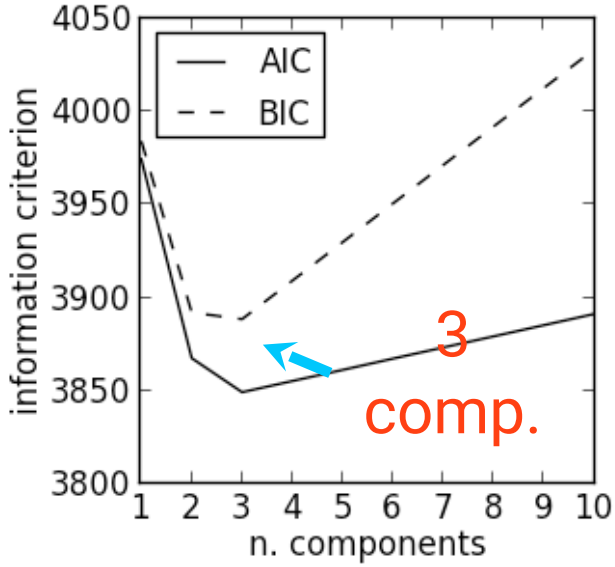
- make this plot by running %run fig_GMM_1D.py

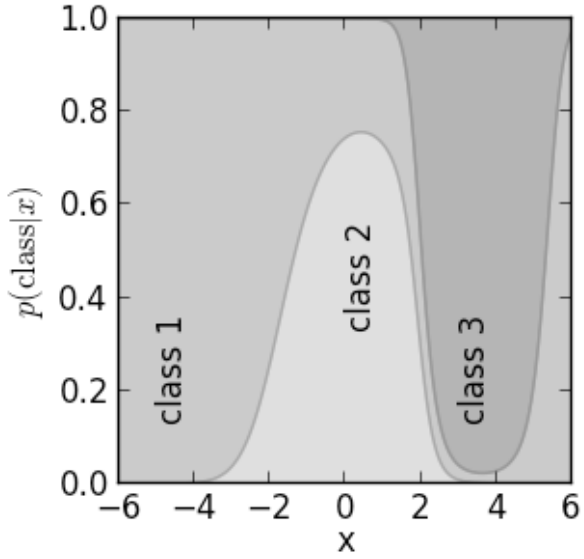
BROKEN – we already know solution!

$$p(j|x_i) = \frac{\alpha_j \mathcal{N}(\mu_j, \sigma_j)}{\sum_{j=1}^{M} \alpha_j \mathcal{N}(\mu_j, \sigma_j)}$$

This best GMM fit is also density estimation!
And GMM fit can also be sample classification into 3 components!







Uses Expectation Maximization method implemented in scikit-learn

Bayesian Information Criterion: it tells you how many components data support!

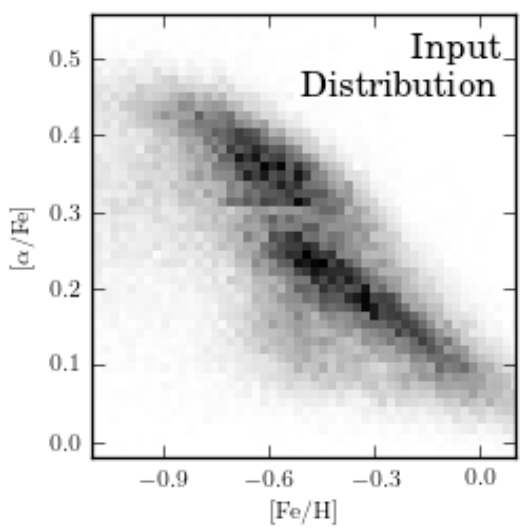
Example of classification for a 3-component mixture

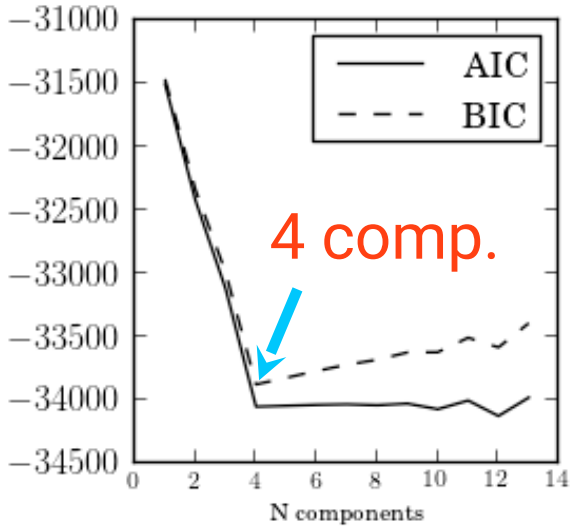
Clustering with Gaussian Mixture in 2D

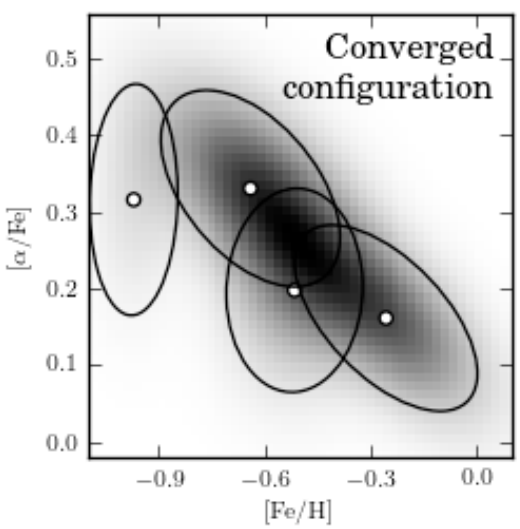
- make this plot by running (it takes a minute or so)
  %run fig_EM_metallicity.py

BROKEN?

This best GMM fit is also density estimation!
And GMM fit can also be used for sample classification into 4 components!







Uses Expectation Maximization method implemented in scikit-learn

Bayesian Information Criterion: it tells you how many components data support!

Example of 2D clustering into a 4-component mixture

Clustering with Gaussian Mixture in 2D

- we may need many components (here 100); data should tell us how many...

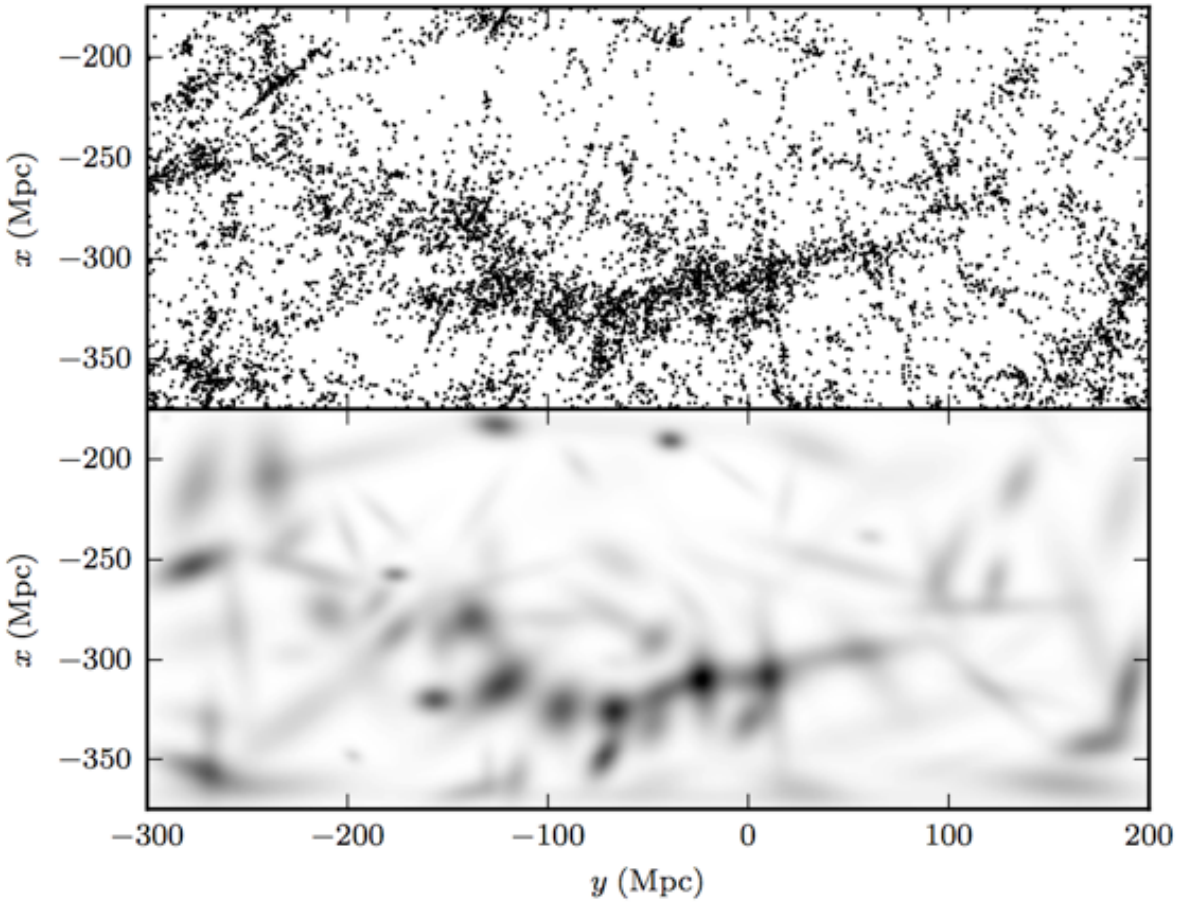


Figure 6.7.: A two-dimensional mixture of 100 Gaussians (bottom) used to estimate the number density distribution of galaxies within the SDSS Great Wall (top). Compare to figures 6.4 and 6.3, where the density for the same distribution is computed using both kernel density and nearest-neighbor-based estimates.

Clustering with Gaussian Mixture in 2D

the ability to recognize structure in the data improves with the dataset size

Simulated data has 4 components (2 narrow and 2 wide); need at least 1000 points to recognize at least 3 components

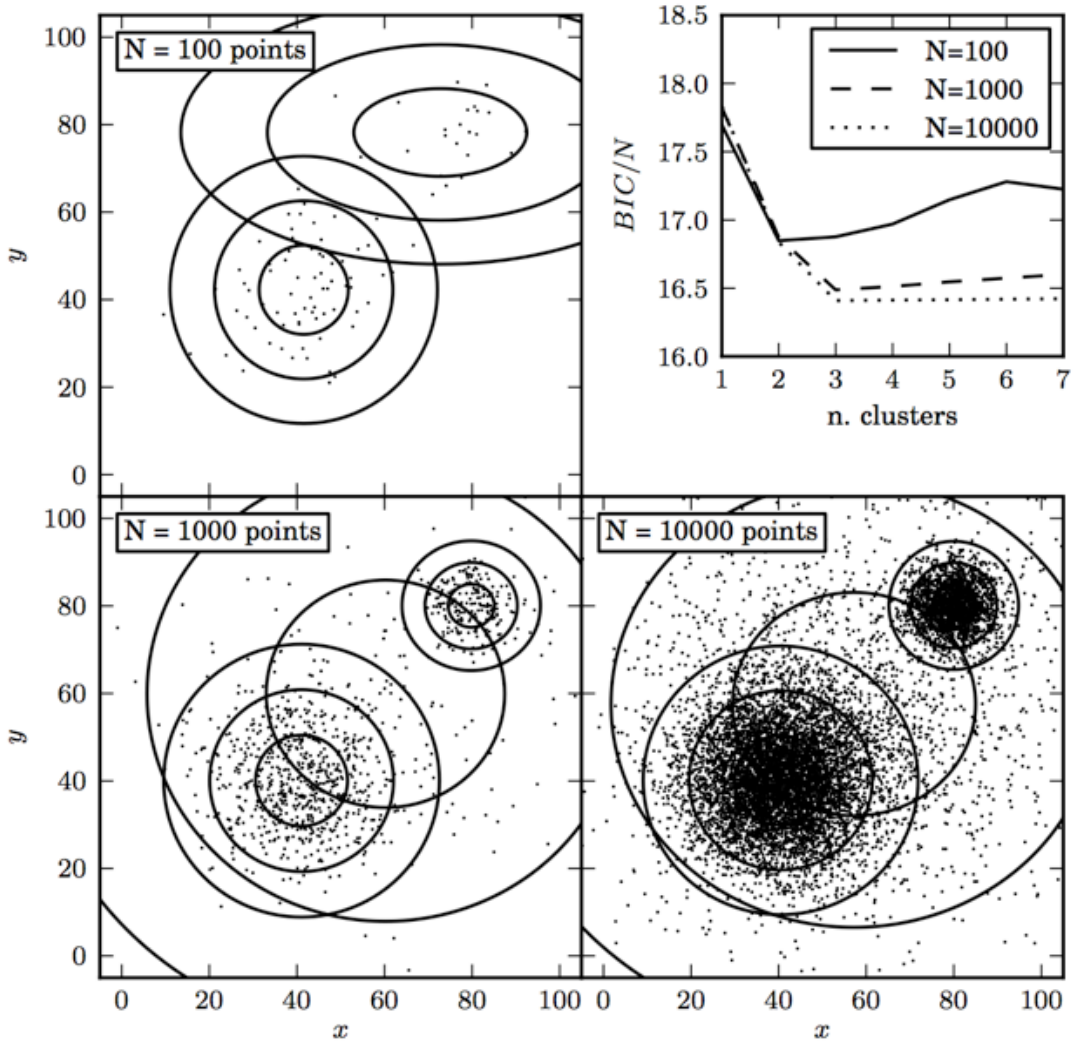


Figure 6.9.: The BIC-optimized number of components in a Gaussian mixture model as a function of the sample size. All three samples (with 100, 1000, and 10000 points) are drawn from the same distribution: two narrow foreground Gaussians and two wide background Gaussians. The top-right panel shows the BIC as a function of the number of components in the mixture. The remaining panels show the distribution of points in the sample and the 1, 2, and 3 standard deviation contours of the best-fit mixture model.

6.4.2. Clustering by sum-of-squares minimization: K-means

One of the simplest methods for partitioning data into a small number of clusters is K-means. K-means seeks a partitioning of the points into K disjoint subsets C_k with each subset containing N_k points such that the following sum-of-squares objective function is minimized:

$$\sum_{k=1}^{K} \sum_{i \in C_k} ||x_i - \mu_k||^2, \tag{6.28}$$

where $\mu_k = \frac{1}{N_k} \sum_{i \in C_k} x_i$ is the mean of the points in set C_k, and $C(x_i) = C_k$ denotes that the class of x_i is C_k.

The procedure for K-means is to initially choose the centroid, μ_k, of each of the K clusters. We then assign each point to the cluster that it is closest to (i.e., according to $C(x_i) = \arg\min_k ||x_i - \mu_k||$). At this point we update the centroid of each cluster by recomputing μ_k according to the new assignments. The process continues until there are no new assignments.

While a globally optimal minimum cannot be guaranteed, the process can be shown to never increase the sum-of-squares error. In practice K-means is run multiple times with different starting values for the centroids of C_k and the result with the lowest sum-of-squares error is used. K-means can be interpreted as a "hard" version of the EM algorithm for a mixture of spherical Gaussians (i.e., we are assuming with K-means that the data can be described by spherical clusters with each cluster containing approximately the same number of points).

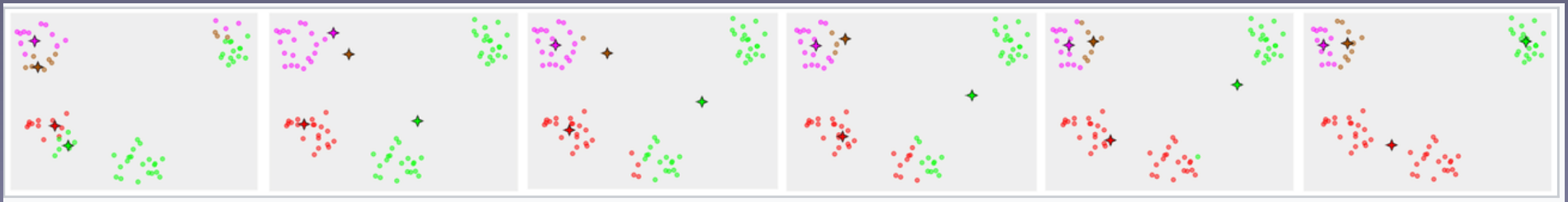
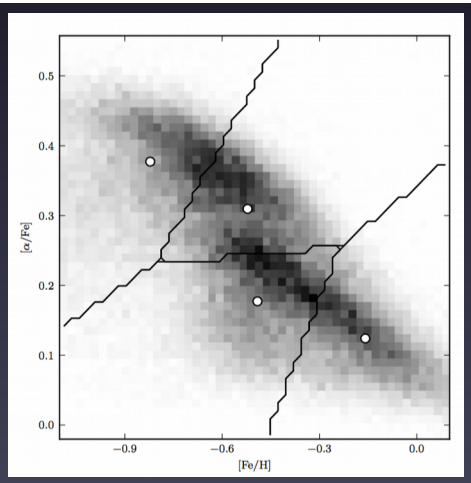
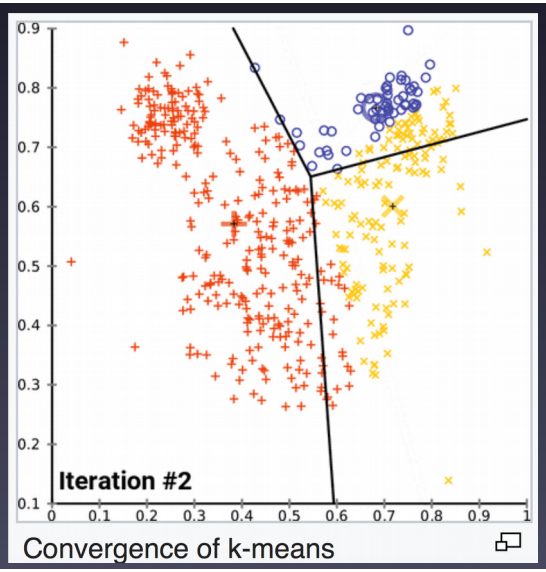


Figure 6.13.: The K-means analysis of the stellar metallicity data used in figure 6.6. Note how the background distribution "pulls" the cluster centers away from the locus where one would place them by eye. This is why more sophisticated models like GMM are often better in practice.

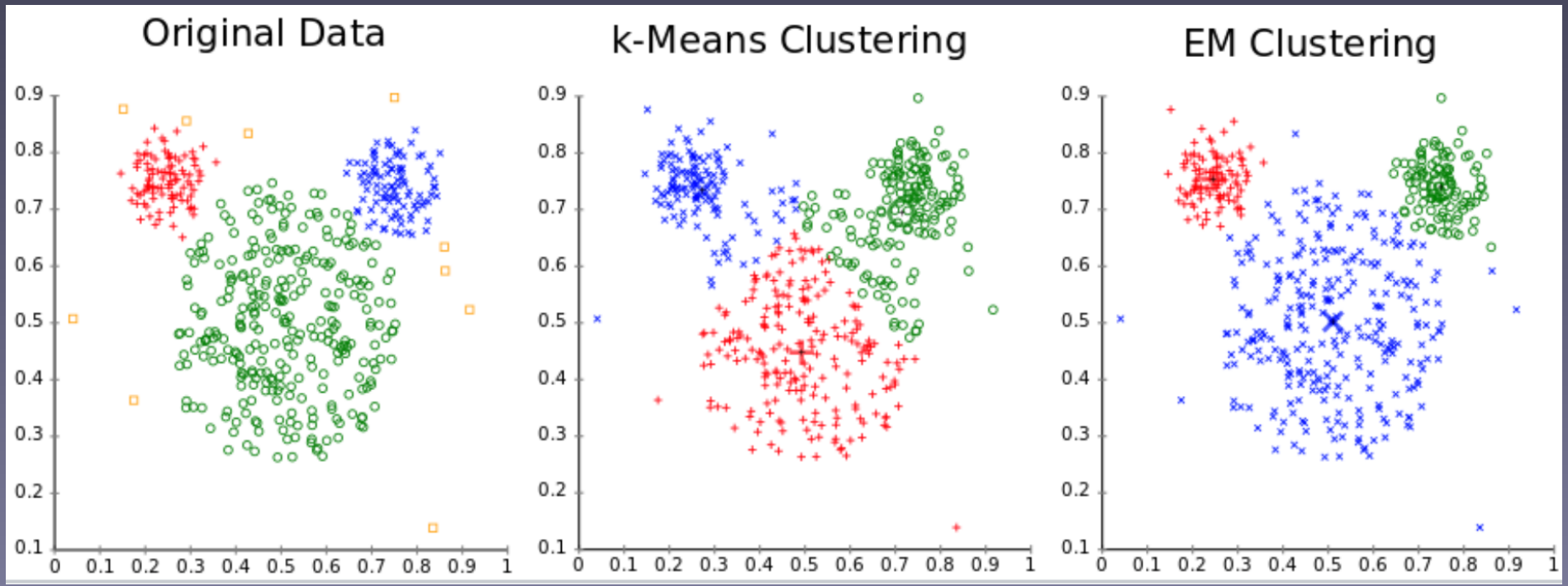
astroML
Figure 6.13





Convergence of k-means

https://en.wikipedia.org/wiki/K-means_clustering

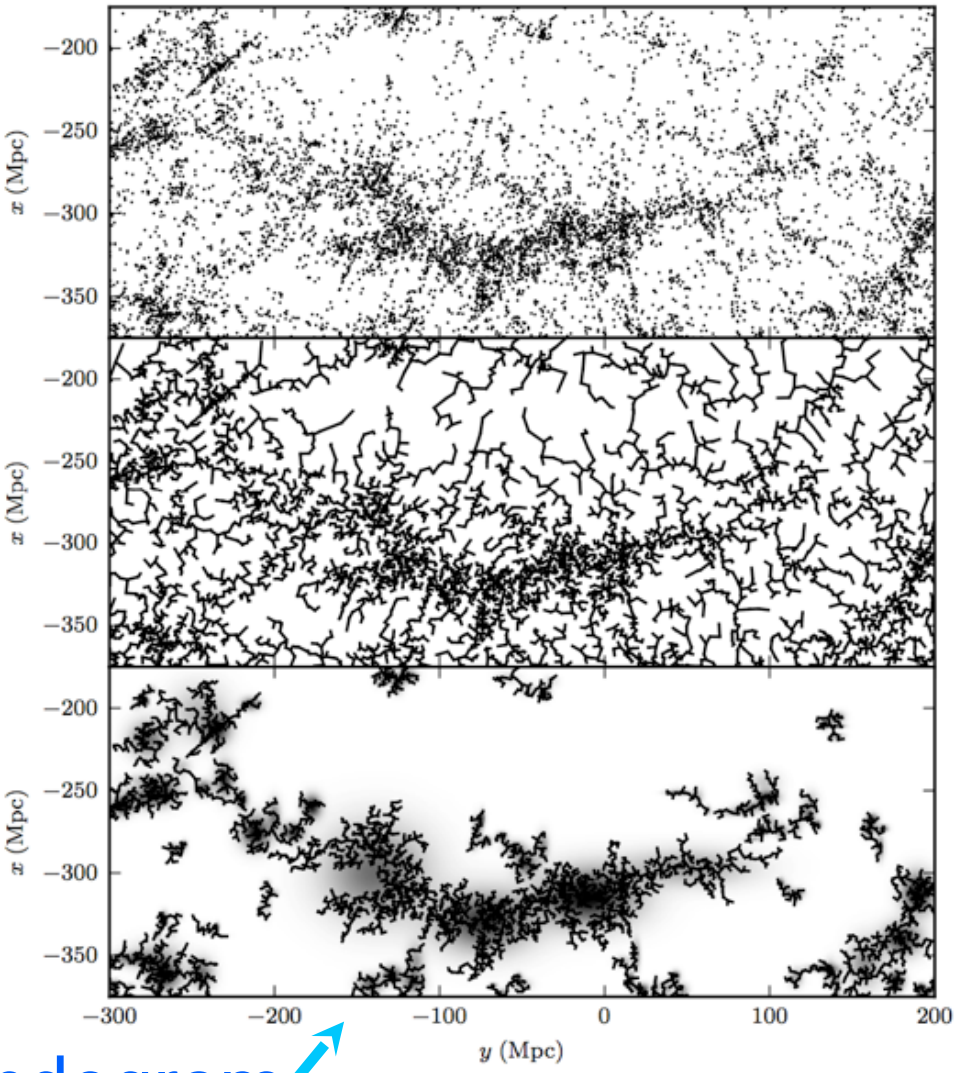


Hierarchical Clustering in 2D

GMM is only one of many clustering algorithms...

SDSS Great Wall again...

Right: clustered using minimum spannig tree algorithm: at each step, nearest clusters are merged...



Dendogram

Figure 6.15.: An approximate Euclidean minimum spanning tree over the two-dimensional projection of the SDSS Great Wall. The upper panel shows the input points, and the middle panel shows the dendrogram connecting them. The lower panel shows clustering based on this dendrogram, created by removing the largest 10% of the graph edges, and keeping the remaining connected clusters with 30 or more members.

Hierarchical Clustering in 2D

Hierarchical clustering can be approached as a top-down (*divisive*) procedure, where we progressively subdivide the data, or as a bottom-up (*agglomerative*) procedure, where we merge the nearest pairs of clusters. For our examples below we will consider the agglomerative approach.

At each step in the clustering process we merge the "nearest" pair of clusters. Options for defining the distance between two clusters, C_k and $C_{k'}$, include

$$d_{\min}(C_k, C_{k'}) = \min_{x \in C_k, x' \in C_{k'}} ||x - x'||, \tag{6.34}$$

$$d_{\max}(C_k, C_{k'}) = \max_{x \in C_k, x' \in C_{k'}} ||x - x'||, \tag{6.35}$$

$$d_{\text{avg}}(C_k, C_{k'}) = \frac{1}{N_k N_{k'}} \sum_{x \in C_k} \sum_{x' \in C_{k'}} ||x - x'||, \tag{6.36}$$

$$d_{\text{cen}}(C_k, C_{k'}) = ||\mu_k - \mu_{k'}||, \tag{6.37}$$

where x and x' are the points in cluster C_k and $C_{k'}$ respectively, N_k and $N_{k'}$ are the number of points in each cluster, and μ_k and $\mu_{k'}$ the centroid of the clusters.

Using the distance $d_{\min}$ results in a hierarchical clustering known as a minimum spanning tree (see [1; 4; 20], for some astronomical applications) and will commonly produce clusters with extended chains of points. Using $d_{\max}$ tends to produce hierarchical clustering with compact clusters. The other two distance examples have behavior somewhere between these two extremes.

Naively, an $O(N^3)$ algorithm, recently (2010) $O(N*\log N)$ shown to be possible.

Outline

- Clustering (unsupervised classification)
  - 1-D hypothesis testing
  - clustering with Gaussian Mixture models (GMM)
  - K-means clustering algorithm
  - hierarchical clustering algorithm
- (supervised) Classification
  - potpourri of supervised classification methods:

    naive Bayes, quadratic discriminant analysis, GMM, KNN, support vector machines
  - classification comparison with ROC curves

Clustering (unsupervised classification)

Here “unsupervised” means that there is no prior information about the number and properties of clusters.

“Clustering” in astronomy refers to a number of different aspects of data analysis.

Given a multivariate point data set, we can ask whether it displays any structure, that is, concentrations of points.

Alternatively, when a density estimate is available we can search for “overdensities.”

Another way to interpret clustering is to seek a partitioning or segmentation of data into smaller parts according to some criteria.

Supervised classification

Here “supervised” means that there is prior information about the number and properties of clusters: for a training sample, we know the so-called “class labels” (for each data point in the training sample, we know to which cluster it belongs; characterizing these known clusters is typically easier than finding and characterizing unknown clusters)

There are two basic types of classification methods: **generative classification** methods model the underlying density field (i.e. it relies on density estimation methods), and **discriminative classification** methods which focus on finding the decision boundary which separates classes directly. The former are easier to interpret, the latter often work better in high-D cases.

Generative classification

Given a set of data $\{\mathbf{x}\}$ consisting of N points in D dimensions, such that x_i^j is the jth feature of the ith point, and a set of discrete labels $\{y\}$ drawn from K classes, with values y_k, Bayes' theorem describes the relation between the labels and features:

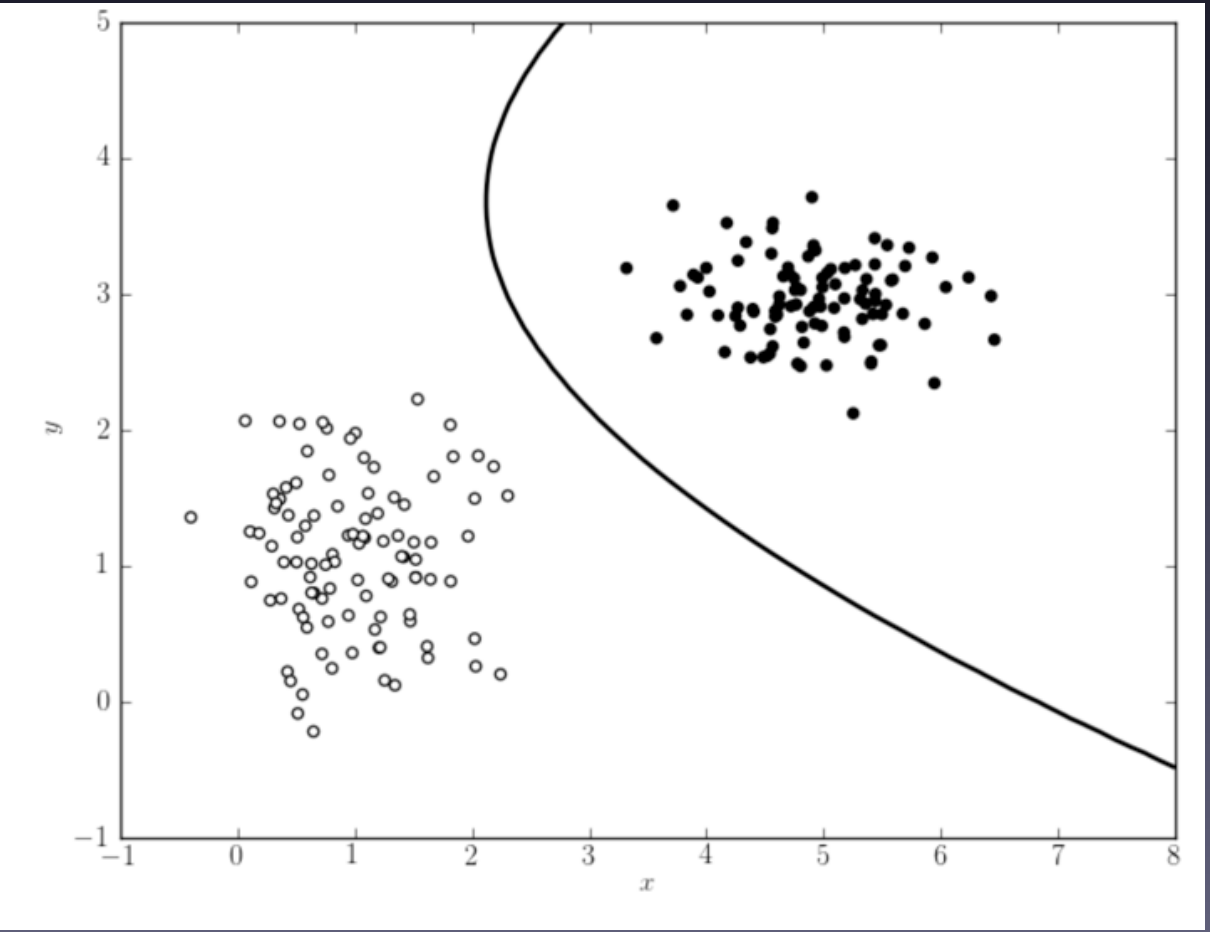
$$p(y_k|\mathbf{x}_i) = \frac{p(\mathbf{x}_i|y_k)p(y_k)}{\sum_i p(\mathbf{x}_i|y_k)p(y_k)}. \tag{9.6}$$

If we knew the full probability densities $p(\mathbf{x}, y)$ it would be straightforward to estimate the classification likelihoods directly from the data. If we chose not to fully sample $p(\mathbf{x}, y)$ with our training set we can still define the classifications by drawing from $p(y|\mathbf{x})$ and comparing the likelihood ratios between classes (in this way we can focus our labeling on the specific, and rare, classes of source rather than taking a brute-force random sample).

In generative classifiers we are modeling the class-conditional densities explicitly, which we can write as $p_k(\mathbf{x})$ for $p(\mathbf{x}|y = y_k)$, where the class variable is, say, $y_k = 0$ or $y_k = 1$. The quantity $p(y = y_k)$, or π_k for short, is the probability of any point having class k, regardless of which point

The task of learning the best classifier then becomes the task of estimating the p_k's. This approach means we will be doing multiple separate *density estimates* using many of the techniques introduced

Discriminative classification



$$\text{completeness} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}},$$

$$\text{contamination} = \frac{\text{false positives}}{\text{true positives} + \text{false positives}}$$

In this simple example, perfect separation is possible; generally not the case!

Discriminative classification

discriminant function

$$g(\mathbf{x}) = f(y|\mathbf{x}) = \int y\, p(y|\mathbf{x})\, dy \tag{9.7}$$

$$= 1 \cdot p(y=1|\mathbf{x}) + 0 \cdot p(y=0|\mathbf{x}) = p(y=1|\mathbf{x}). \tag{9.8}$$

If we now apply Bayes' rule (eq. 3.10), we find (cf. eq. 9.6)

$$g(\mathbf{x}) = \frac{p(\mathbf{x}|y=1)\, p(y=1)}{p(\mathbf{x}|y=1)\, p(y=1) + p(\mathbf{x}|y=0)\, p(y=0)} \tag{9.9}$$

$$= \frac{\pi_1 p_1(\mathbf{x})}{\pi_1 p_1(\mathbf{x}) + \pi_0 p_0(\mathbf{x})}. \tag{9.10}$$

Bayes classifier

Making the discriminant function yield a binary prediction gives the abstract template called a *Bayes classifier*. It can be formulated as

$$\hat{y} = \begin{cases} 1 & \text{if } g(\mathbf{x}) > 1/2, \\ 0 & \text{otherwise,} \end{cases} \tag{9.11}$$

$$= \begin{cases} 1 & \text{if } p(y=1|\mathbf{x}) > p(y=0|\mathbf{x}), \\ 0 & \text{otherwise,} \end{cases} \tag{9.12}$$

$$= \begin{cases} 1 & \text{if } \pi_1 p_1(\mathbf{x}) > \pi_0 p_0(\mathbf{x}), \\ 0 & \text{otherwise.} \end{cases} \tag{9.13}$$

This is easily generalized to any number of classes K, since we can think of a $g_k(\mathbf{x})$ for each class (in a two-class problem it is sufficient to consider $g(\mathbf{x}) = g_1(\mathbf{x})$). The Bayes classifier is a template in the sense that one can plug in different types of model for the p_k's and the π's.

Decision boundary

The *decision boundary* between two classes is the set of x values at which each class is equally likely; that is,

$$\pi_1 p_1(\mathbf{x}) = \pi_2 p_2(\mathbf{x}); \tag{9.14}$$

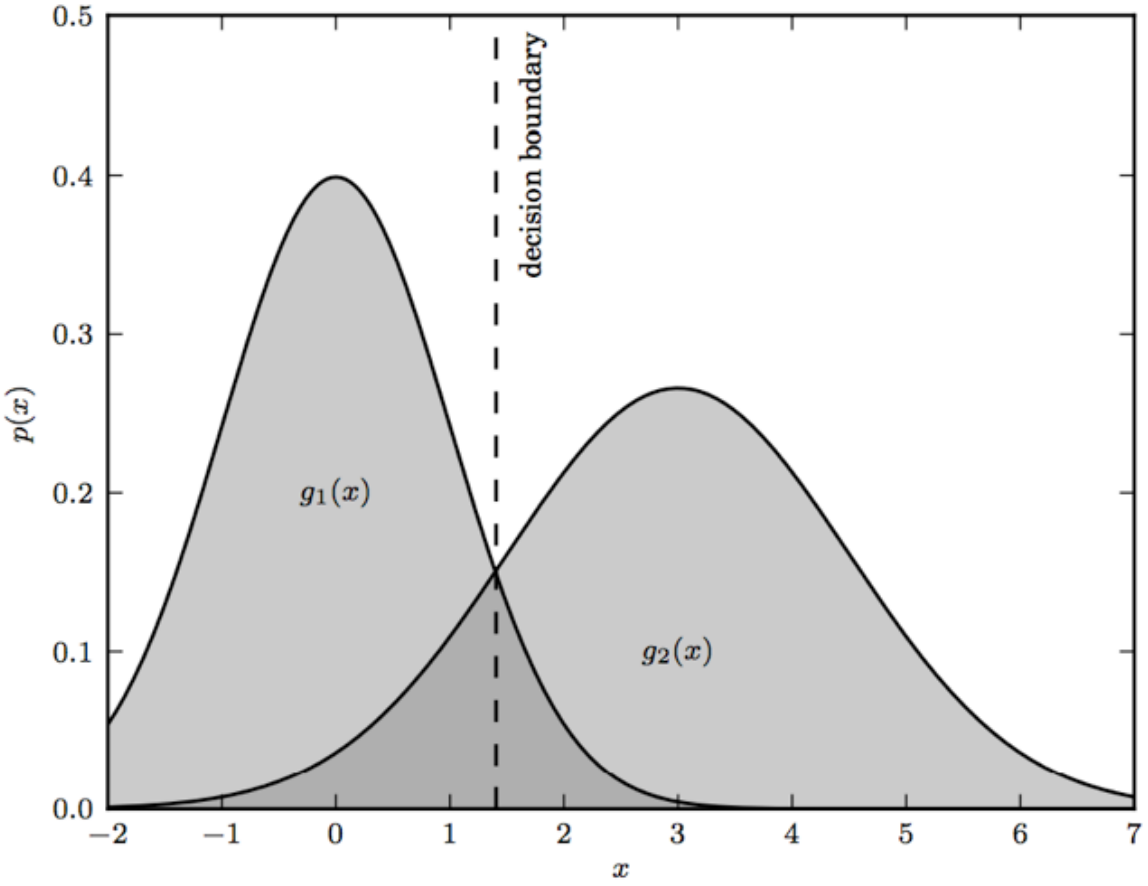
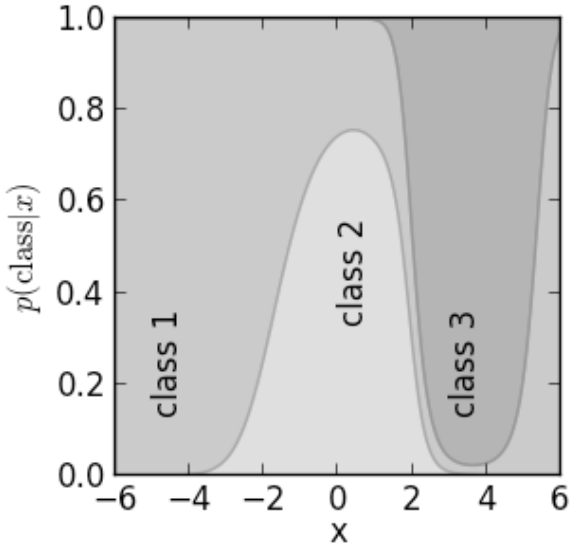


Figure 9.1.: An illustration of a decision boundary between two Gaussian distributions.



Naive Bayes classifier

The Bayes classifier formalism presented above is conceptually simple, but can be very difficult to compute: in practice, the data $\{\mathbf{x}\}$ above may be in many dimensions, and have complicated probability distributions. We can dramatically reduce the complexity of the problem by making the assumption that all of the attributes we measure are conditionally independent. This means that

$$p(x^i, x^j|y_k) = p(x^i|y)p(x^j|y_k), \tag{9.15}$$

$$p(y_k|x^0, x^1, \ldots, x^N) = \frac{\prod_i p(x^i|y_k)p(y_k)}{\sum_j \prod_i p(x^i|y_j)p(y_j)}.$$

prior for a given class!

recall:

$$p(j|x_i) = \frac{\alpha_j \mathcal{N}(\mu_j, \sigma_j)}{\sum_{j=1}^{M} \alpha_j \mathcal{N}(\mu_j, \sigma_j)}$$

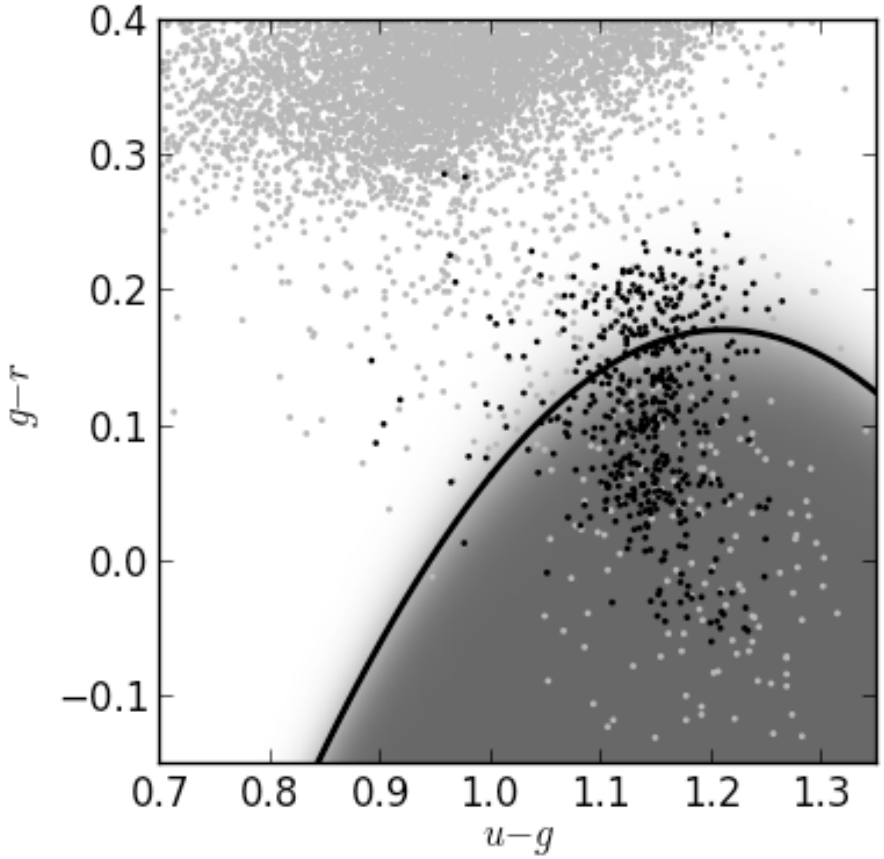
Gaussian naive Bayes classifier assumes that the data can be modeled by a sum of axis-aligned multi-variate gaussians

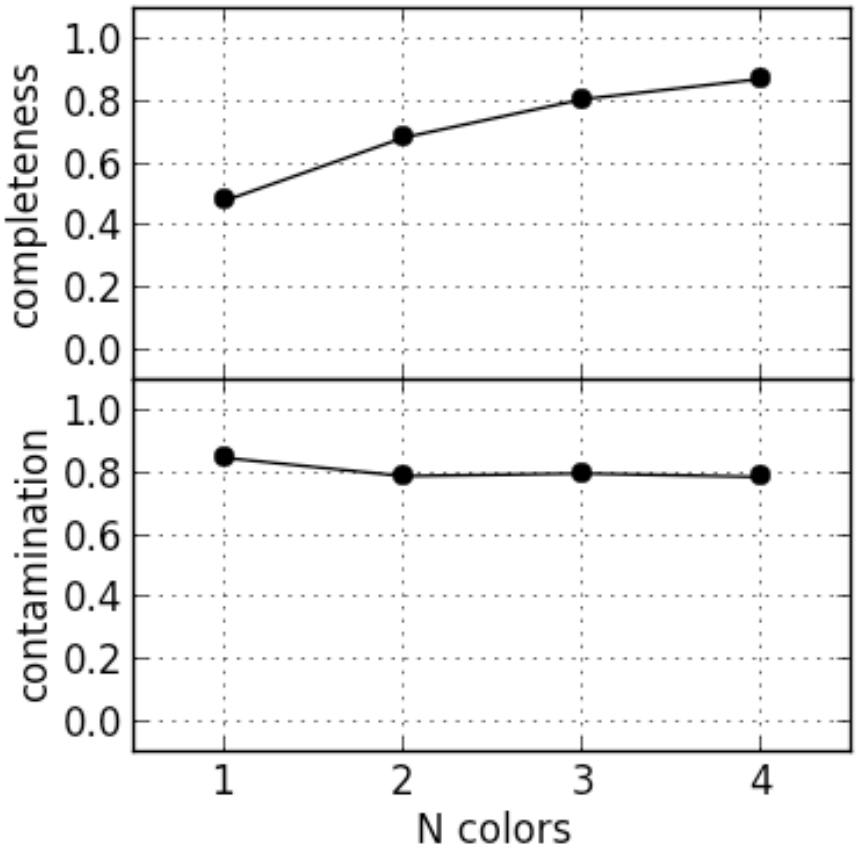
Gaussian naive Bayes classification of RR Lyrae using colors

- make this plot by running

  %run fig_rrlyrae_naivebayes.py

Black dots are 483 RR Lyrae, gray dots are the much larger sample of ~93,000 other stars (for this real-world dataset it is impossible to achieve perfect classification!)

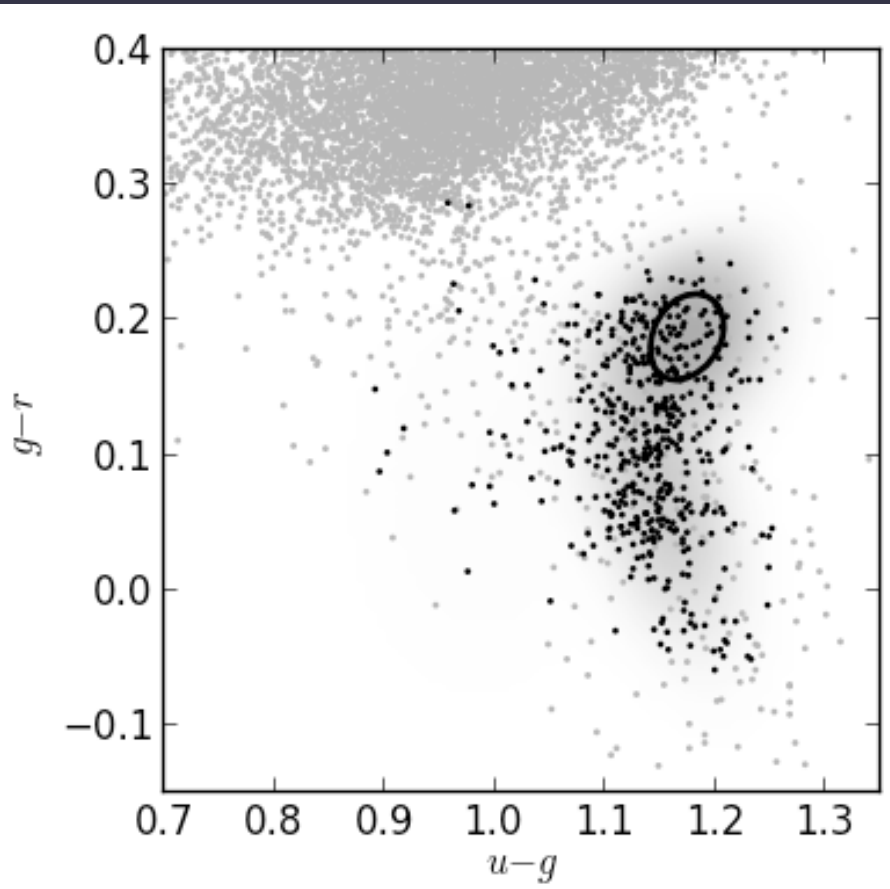


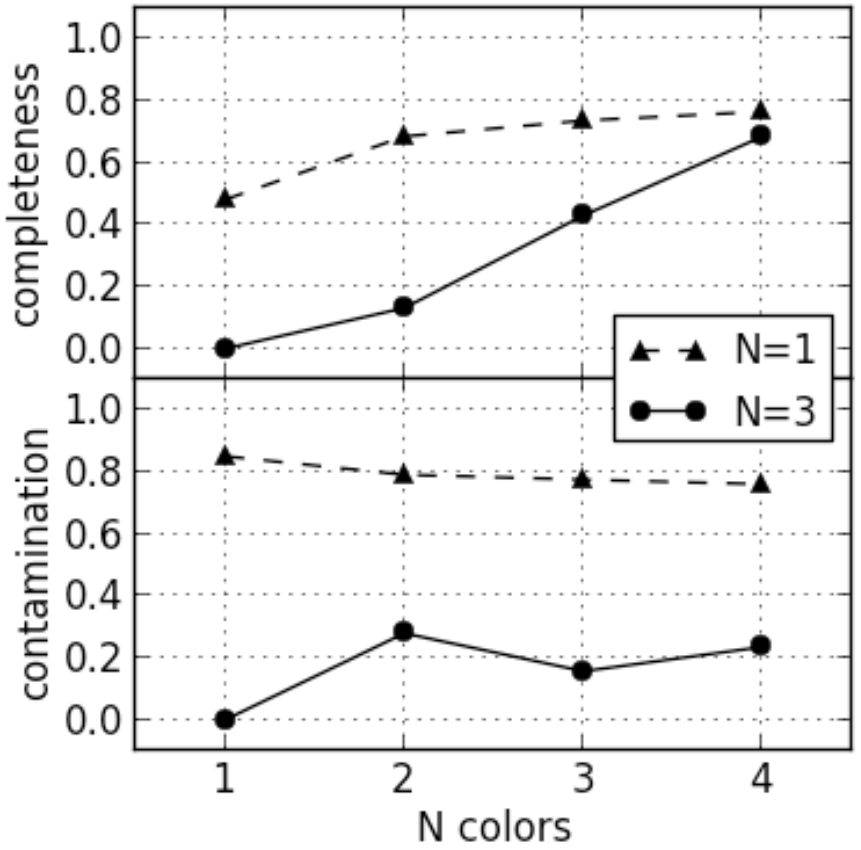


Gaussian Mixture Bayes classification of RR Lyrae

- make this plot by running
  %run fig_rrlyrae_GMMbayes.py

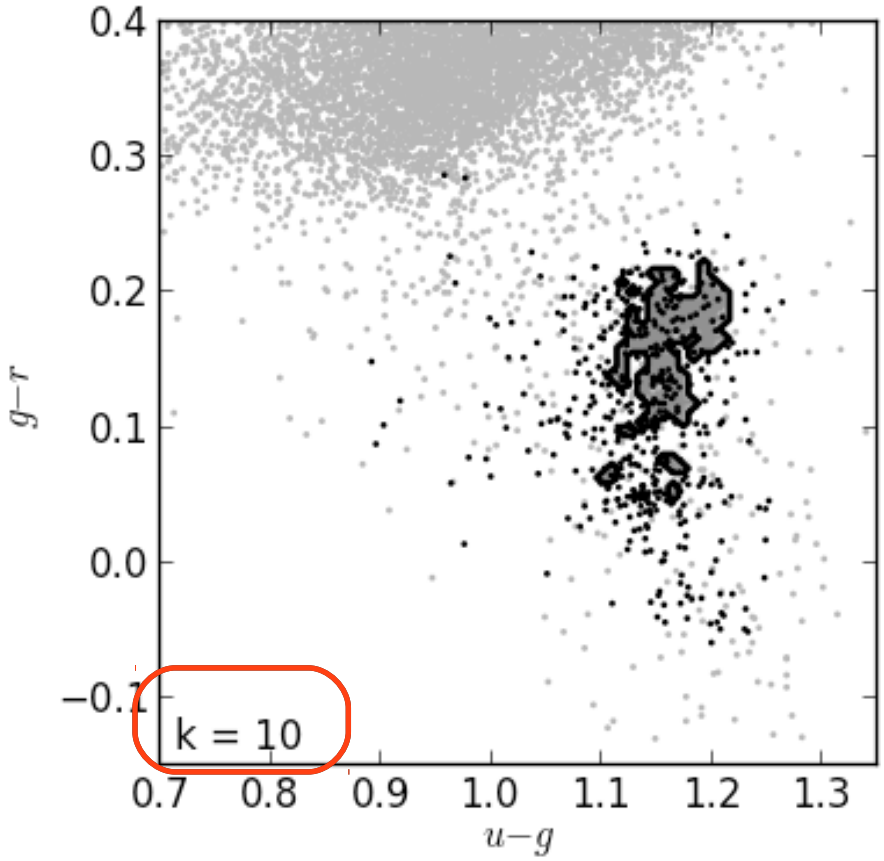
GMM Bayes classifier does **not** assume that gaussian components are aligned with coordinate axes

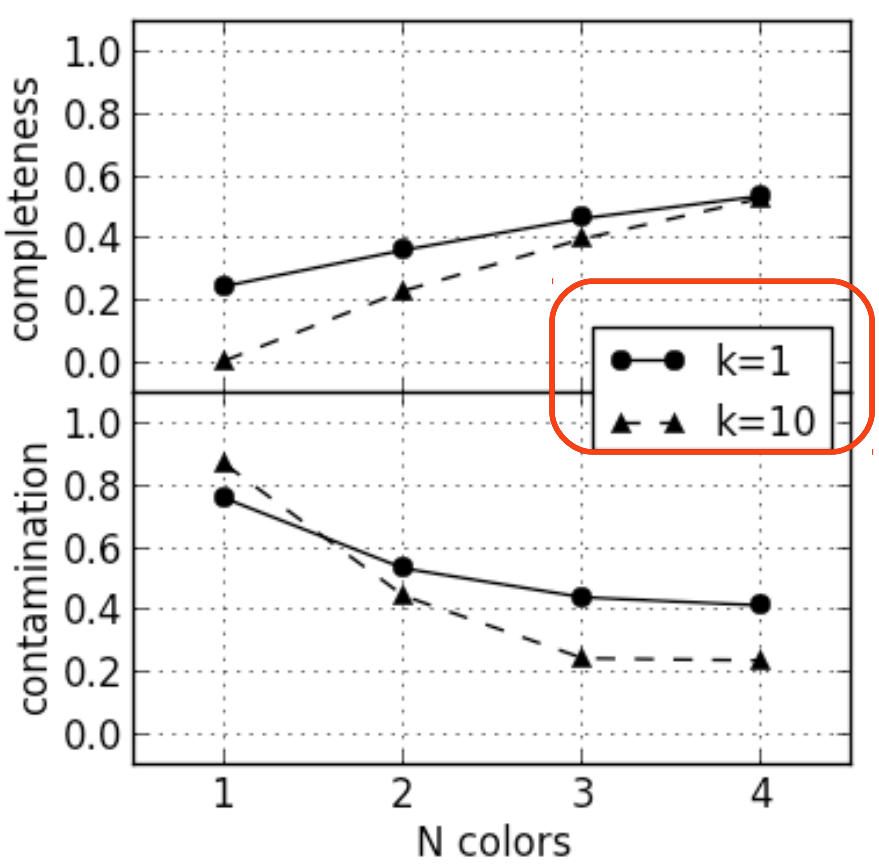




K nearest neighbors classification of RR Lyrae

- make this plot by running
  %run fig_rrlyrae_knn.py





KNN classifier is a non-parametric classifier: labels of K neighbors are weighted by distance (need large samples)

Discriminative classification

With nearest-neighbor classifiers we started to see a subtle transition—while clearly related to Bayes classifiers using variable-bandwidth kernel estimators, the class density estimates were skipped in favor of a simple classification decision. This is an example of *discriminative classification*, where we directly model the decision boundary between two or more classes of source. Recall, for $y \in \{0, 1\}$, the discriminant function is given by $g(x) = p(y = 1|x)$. Once we have it, no matter how we obtain it, we can use the rule

$$\widehat{y} = \begin{cases} 1 & \text{if } g(x) > 1/2, \\ 0 & \text{otherwise,} \end{cases} \tag{9.28}$$

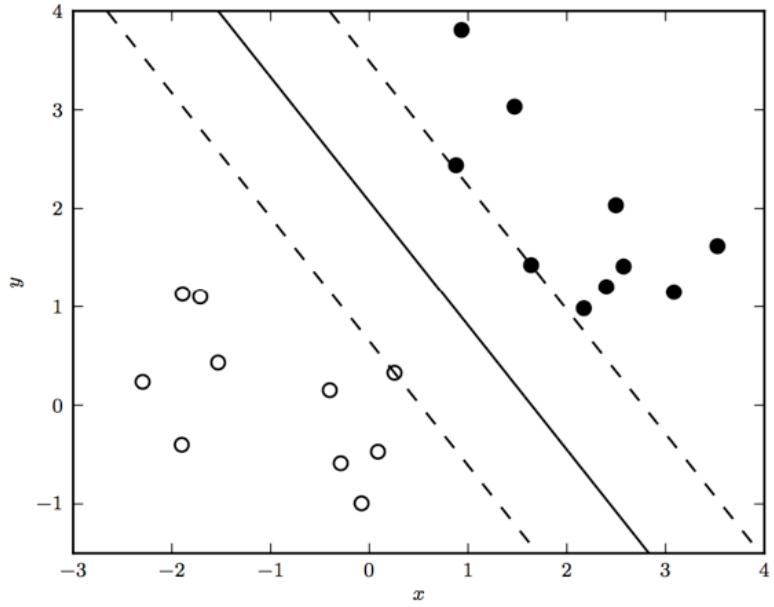


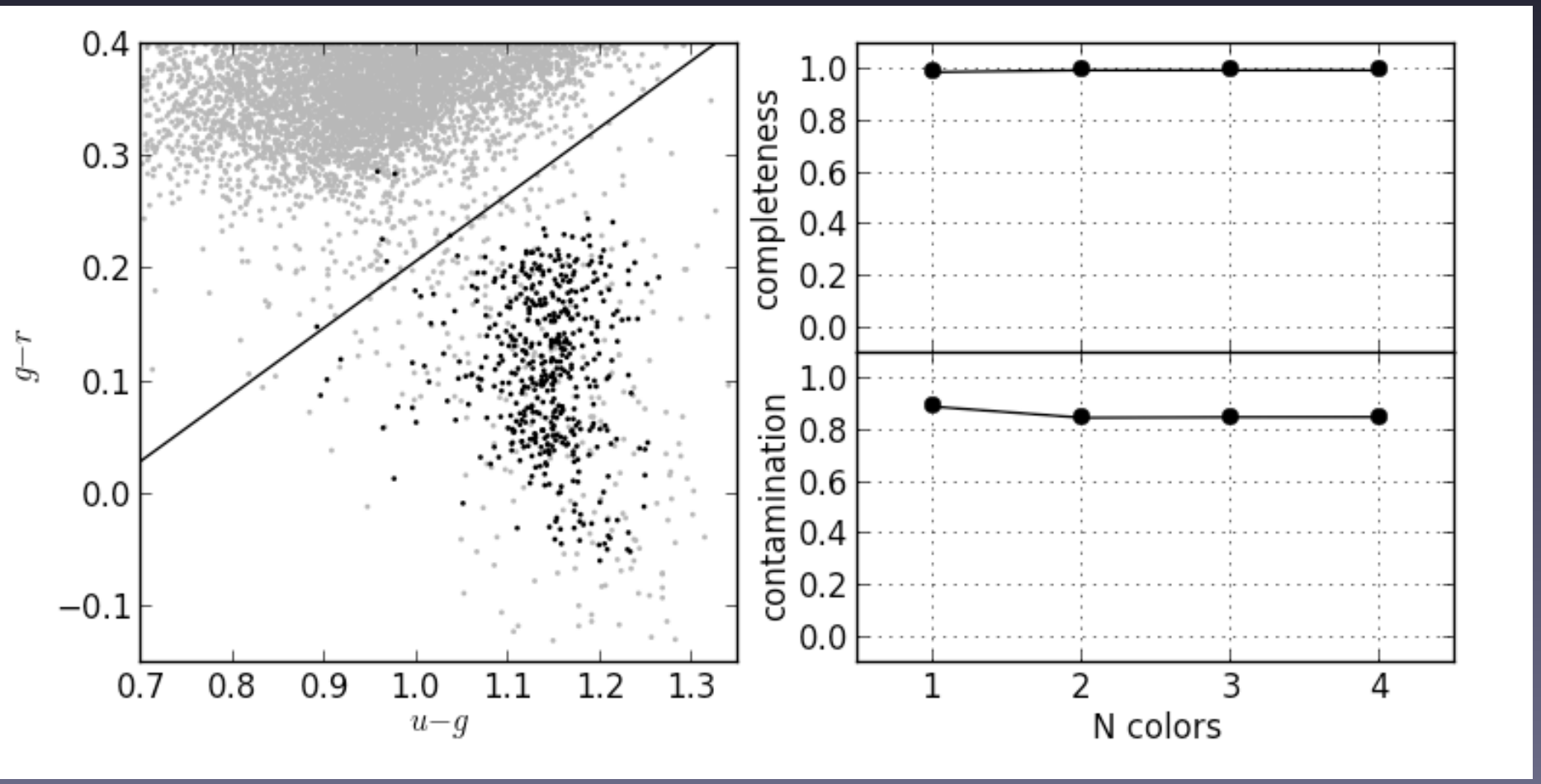
Figure 9.9.: Illustration of SVM. The region between the dashed lines is the *margin*, and the points which the dashed lines touch are called the *support vectors*.

Left: the idea behind the Support Vector Machine classifier: a collection of hyperplanes which maximize the class separation

SVM classification of RR Lyrae

- make this plot by running (it takes a few minutes!)

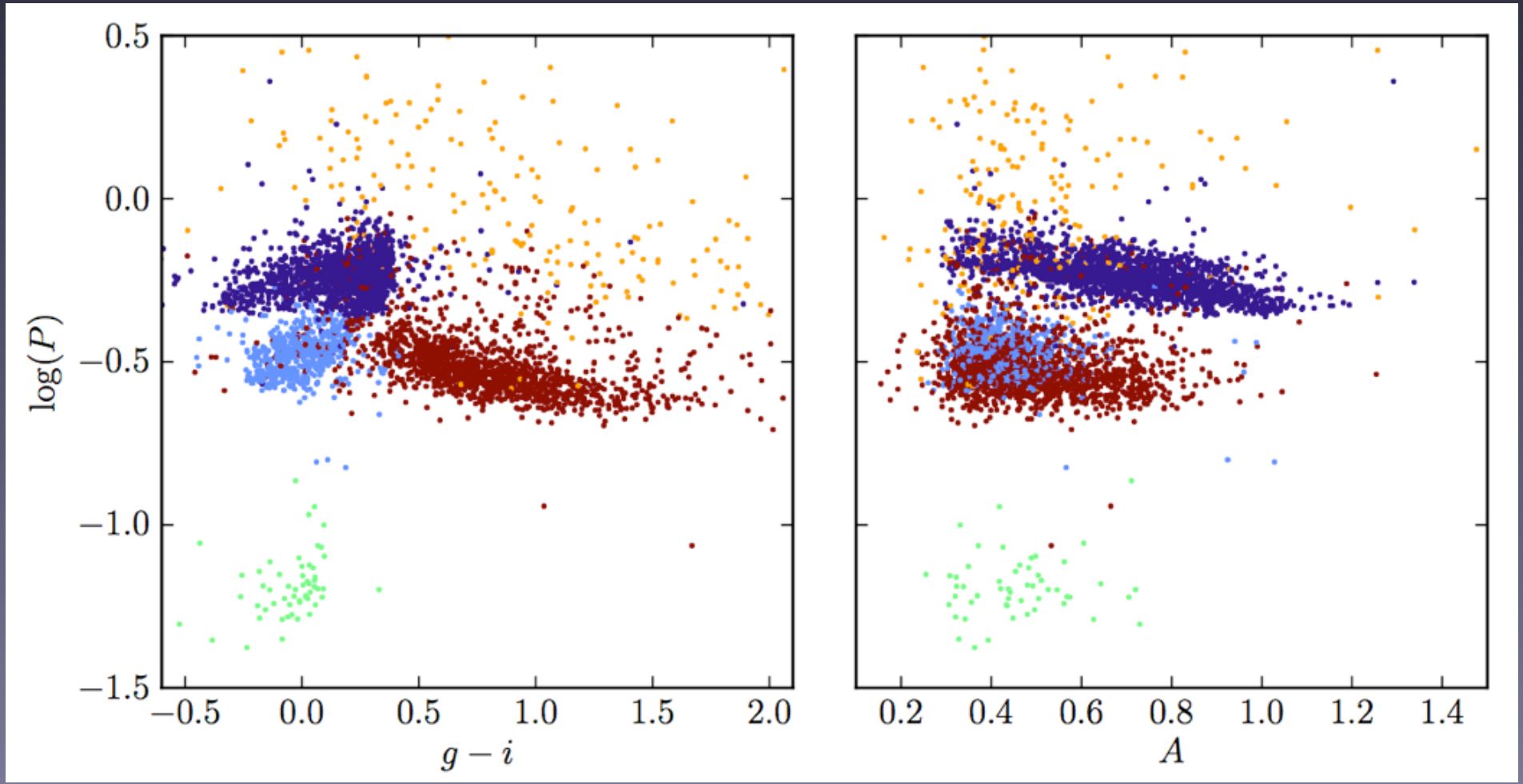
```
%run fig_rrlyrae_svm.py
```



SVM is very robust to outliers; note the very high completeness (at the expense of high contamination)

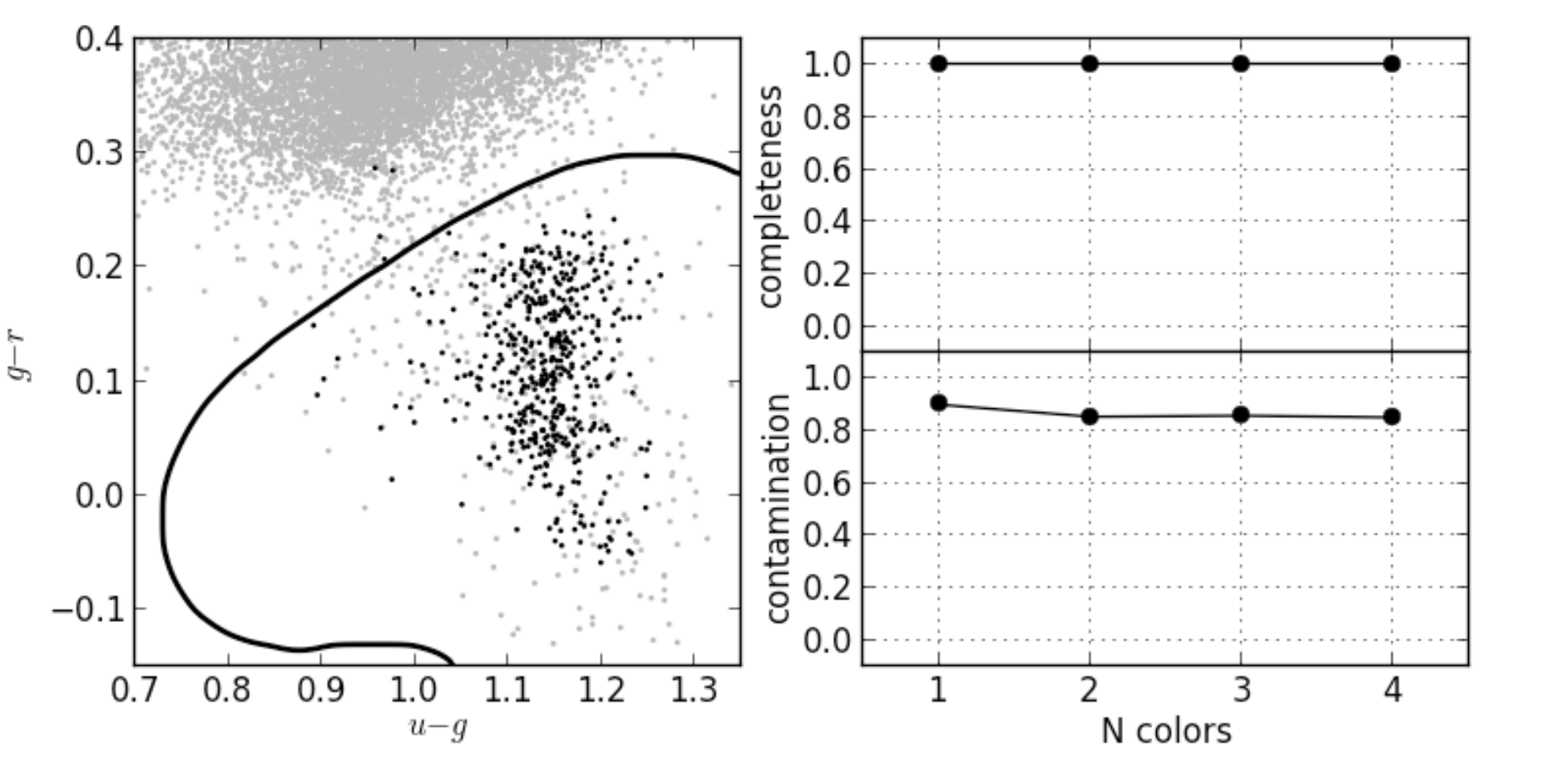
SVM classification of variable stars from Palaversa+ 2013

- Based on 7 attributes (4 SDSS-2MASS colors and 3 light curve parameters: period, amplitude, skewness) and 5 input classes (also in astroML - you can experiment later...)



Kernel SVM classification of RR Lyrae

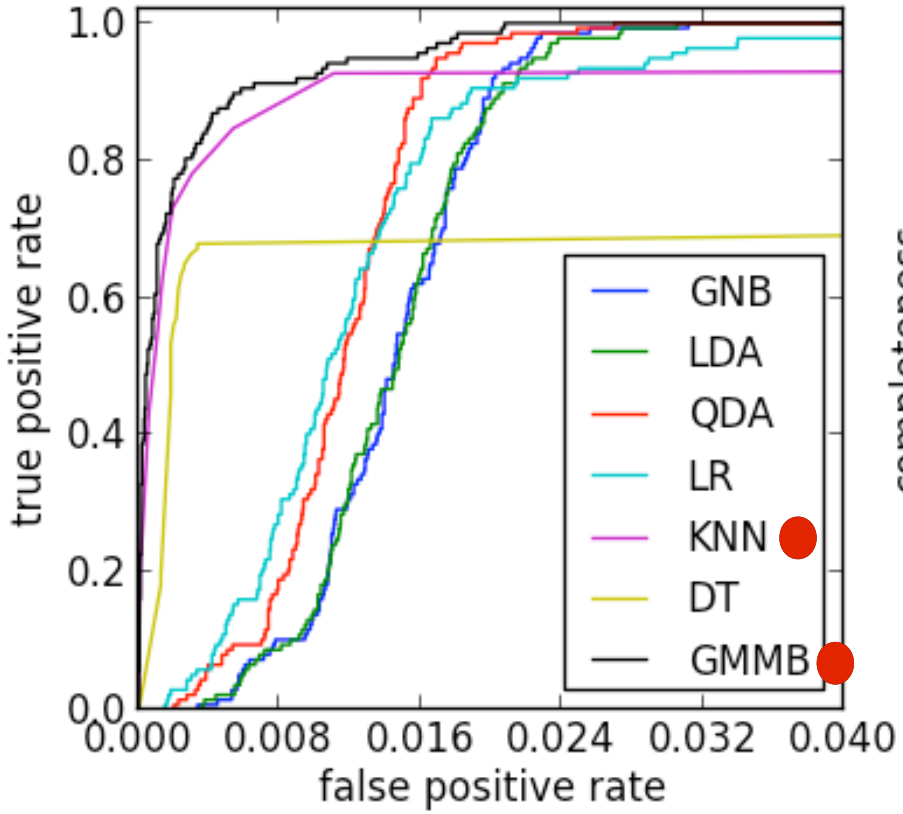
- make this plot by running (it takes a few minutes!)
  %run fig_rrlyrae_kernelsvm.py

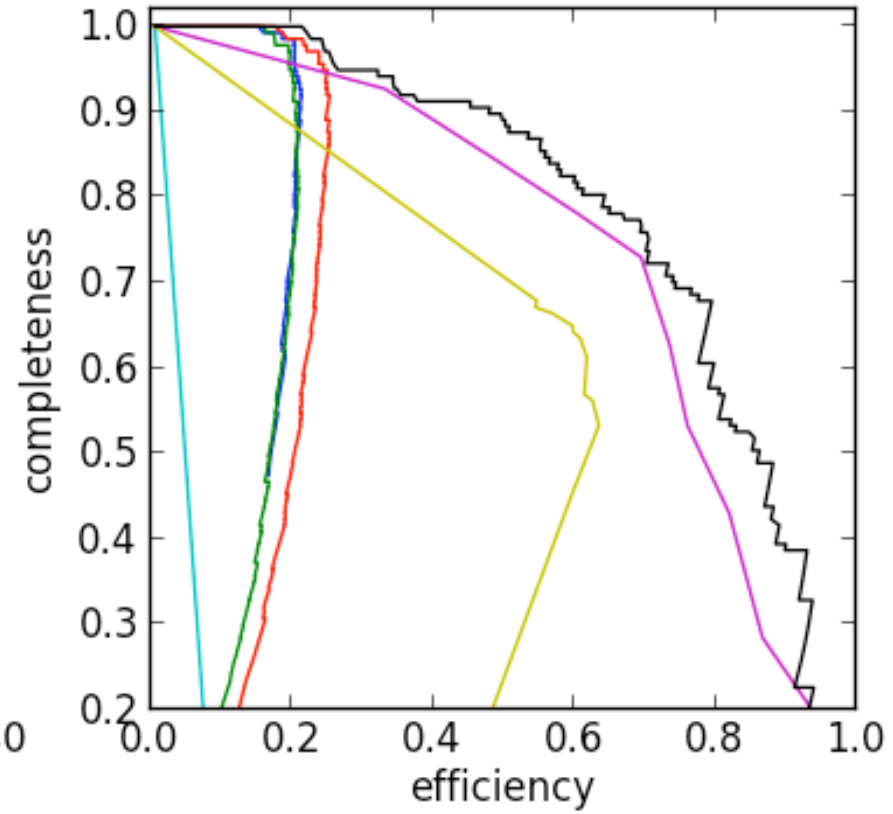


The SVM class decision boundaries do not have to be line
(straight lines and hyperplanes): can use kernel tricks

Comparison of classification methods with ROC curves

- make this plot by running (it takes a few minutes!)
  %run fig_ROC_curve.py

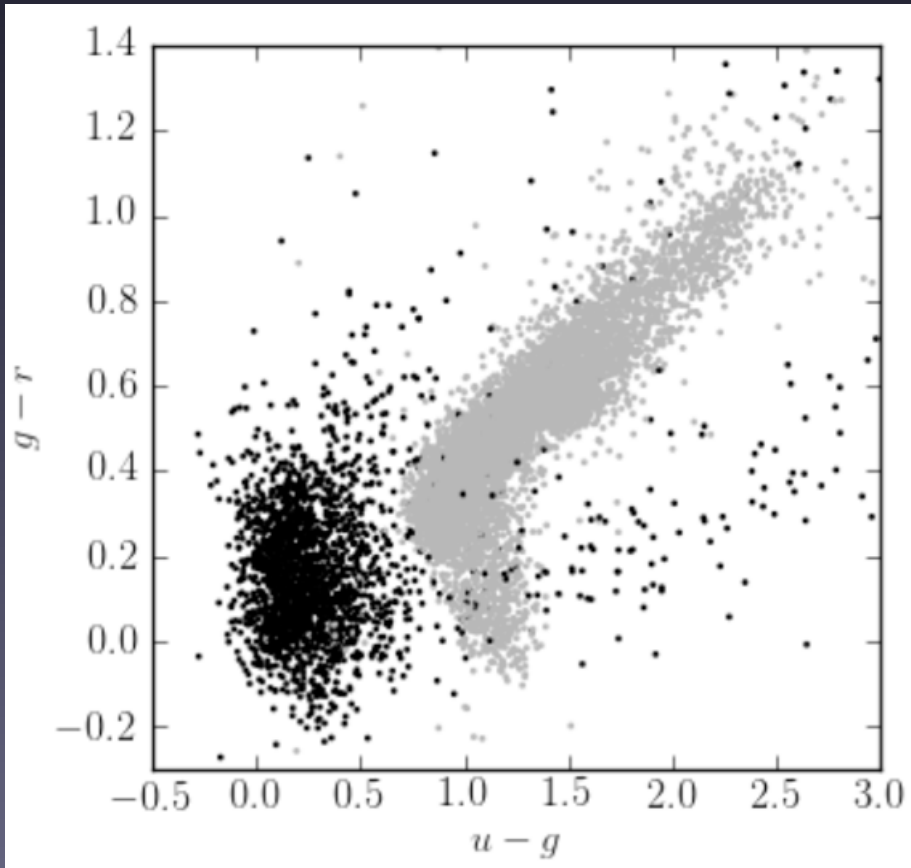


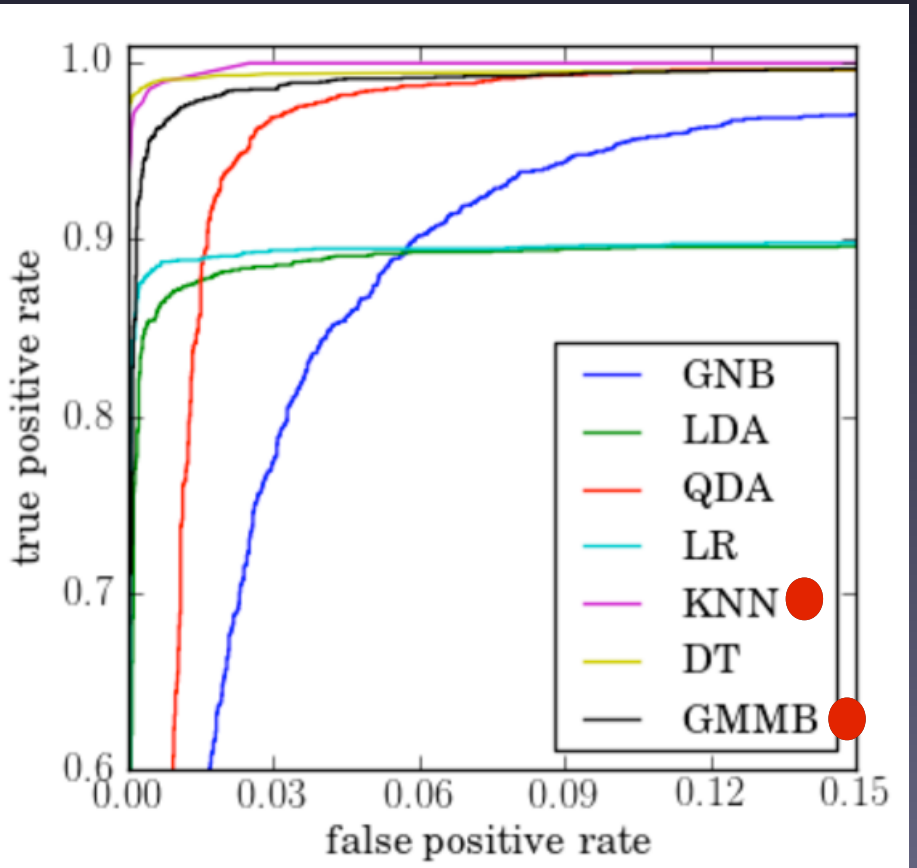


For each method, completeness and contamination can be optimized using ROC curves, and methods can be compared

Comparison of classification methods with ROC curves

- an example of photometric star (gray) vs. quasar (black) separation using SDSS photometry (UV excess method)





Which method would you choose? Why? Would it fail for a different sample? There are many more methods in astroML, and even many many more in the literature...